Tecnológico de Monterrey

Inteligencia artificial avanzada para la ciencia de datos I

Módulo 2
Grupo 101

Roberto Valdez Jasso

A01746863

Análisis y Reporte sobre el desempeño del modelo.

Estado de México

Lunes 11 de septiembre de 2022.

Descripción:

1. Escoge una de las 2 implementaciones que tengas y genera un análisis sobre su desempeño en un conjunto de datos. Este análisis lo deberá documentar en un informe con indicadores claros y gráficos comparativos que respalden su análisis.
2. El análisis debe contener los siguientes elementos:
    a. Separación y evaluación del modelo con un conjunto de prueba y un conjunto de validación (Train/Test/Validation).
    b. Diagnóstico y explicación del grado de sesgo o sesgo: bajo medio alto
    c. Diagnóstico y explicación del grado de variación: bajo medio alto
    d. Diagnóstico y explicación del nivel de ajuste del modelo: underfitt fitt overfitt
3. Basándote en lo encontrado en tu análisis utiliza técnicas de regularización o ajuste de parámetros para mejorar el desempeño de tu modelo y documenta en tu informe cómo mejoró este.

Elección:

El código que será analizado para la evaluación del modelo pruebas, validación y de separación de los datos será:

- Primera implantación de Aprendizaje maquina sin uso de librerías con el data set de Titanic real.

Proceso:

Tras haber elegido el modelo realizado para la actividad de aprendizaje maquina sin el uso de librerías con el data set y con los datos del Titanic original fue el siguiente:

Al momento de realizar generar la separación y limpieza de la data set del titanic una vez ya implementado fue primero saber con que tipos de datos estamos trabajando para denotar con que estamos trabajando:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

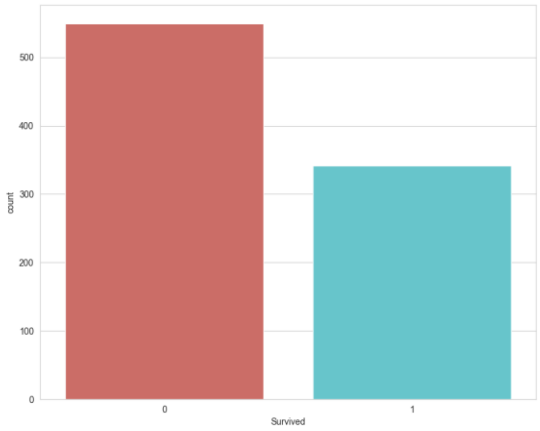
En si la gran mayoría de los datos con los que estamos trabajando en sus columnas, son numéricos o flotantes, mientras que los otros que son objectos , son strings que cuenta con la descripción, nombre, cabina, donde embarco entre otros datos relevantes en el mismo, específicamente lo siguiente:

```
"""
Descripcion de las variables

Survived: Survival (0 = NO, 1= YES) (variable binaria)
pClass = Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
Name = Name
Sex = Sex
Age = Age
SibSp = Number of Siblings/Spouses Aboard
Parch = Number of Parents/Chindren Aboard
Ticket = Number of Tickets
Fare = Passanger Fare (British Pound)
Cabin = Cabin
Embarked = Port of Embarkation ( C = Cherbour ,France;
                                 Q = Queenstown, UK;
                                 S = Southampton, Cobh- Ireland;

                                ) variable categorica
"""
```

Después de saber con qué estamos trabajando toca preparar los datos con los que vamos a trabajar. para preparar los datos primero necesitamos checar si la variable target (Survived)es binaria Queremos comprobar, probar y predecir la sobrevivencia, pero primero hay que comprobar si es binario la cual lo denotamos con la siguiente grafica



Con la gráfica anterior podemos denotar que la sobrevivencia en efecto es una variable binaria.

Ahora checamos si hay valores perdidos. Siempre hay que hacerlo para poder checar que las columnas a las que les falta y como dar a un acercamiento a esos valores perdidos.

```
titanic_training.isnull()
sum = titanic_training.isnull().sum()
sum
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Tambien, checamos las estadísticas descriptivas incluyen aquellas que resumen la tendencia central, la dispersión y la forma de la distribución de un conjunto de datos, excluyendo los valores de NaN., la cual lo denotamos de la siguiente manera:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

En base a estos valores perdidos los valores perdidos y sus revisiones estadísticas podemos que sigamos adelante y eliminemos todas las variables que no son relevantes para predecir la supervivencia. Al menos deberíamos mantener lo siguiente:

- Sobrevivido - Esta variable es obviamente relevante.
- Pclass: ¿la clase de un pasajero en el barco afecta su capacidad de supervivencia?
- Sexo - ¿Podría el género de un pasajero afectar su tasa de supervivencia?
- Edad: ¿la edad de una persona afecta su tasa de supervivencia?
- SibSp: ¿el número de parientes en el barco (que son hermanos o cónyuge) afecta la capacidad de supervivencia de una persona? Probabilidad
- Parch - ¿El número de parientes en el barco (que son niños o padres) afecta la capacidad de supervivencia de una persona? Probabilidad
- Tarifa: ¿la tarifa que pagó una persona afecta su capacidad de supervivencia? Tal vez - vamos a mantenerlo.
- Embarcado - ¿Importa el punto de embarque de una persona? Depende de cómo se llenó el bote... Quedémoslo

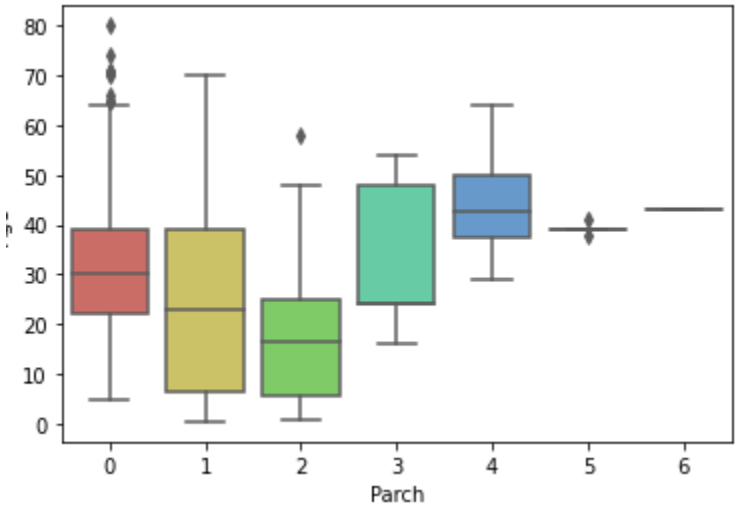
¿Qué sucede con el nombre de una persona, el número de boleto y el número de identificación del pasajero? Son irrelevantes para predecir la capacidad de supervivencia.

Y tambien en la variable de la cabina faltan casi todos los valores, por lo que podemos eliminarlos todos.

```
# Generamos un nuevo dataframe con la elimanacion de los campos anteriores
titanic_data = titanic_training.drop(['Name', 'Ticket', 'Cabin'], axis= 1) # tumbamos las columnas
titanic_data.head() # dataset limpio con datos relevantes
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S

Ya teniendo el data set “limpio”, vamos a checar los valores desaparecidos en base la distribución de los datos en Parch y Age.



Con estas grafica pode decir que hay relacione entre ambas variable, es decir, podemos ver que puede a ver relativos n en base a la edad de del inicial en el bote la cual podemos generar una media con la agrupación de Parch,

```
Parch_groups = titanic_data.groupby(titanic_data['Parch'])
Parch_groups.mean() # Sacamos la media de checar las edades
                        # por categoria tenemos la mediade las edades
                        # para las personas que tenga cero hijos o padres en el barco la edad promedio es de 32
                        # para las personas que tenga un hijo o padre en el barco la edad promedio es de 24
                        # para las personas que tenga dos hijos o padres en el barco la edad promedio es de 17
                        # para las personas que tenga tres hijos o padres en el barco la edad promedio es de 33
                        # para las personas que tenga cuatro hijos o padres en el barco la edad promedio es de 44
                        # para las personas que tenga cinco hijos o padres en el barco la edad promedio es de 39
                        # para las personas que tenga seis hijos o padres en el barco la edad promedio es de 43
```

Parch	PassengerId	Survived	Pclass	Age	SibSp	Fare
0	445.255162	0.343658	2.321534	32.178503	0.237463	25.586774
1	465.110169	0.550847	2.203390	24.422000	1.084746	46.778180
2	416.662500	0.500000	2.275000	17.216912	2.062500	64.337604
3	579.200000	0.600000	2.600000	33.200000	1.000000	25.951660
4	384.000000	0.000000	2.500000	44.500000	0.750000	84.968750
5	435.200000	0.200000	3.000000	39.200000	0.600000	32.550000
6	679.000000	0.000000	3.000000	43.000000	1.000000	46.900000

Lo que se puede decir que por medio de las edades son:

- para las personas que tengan cero hijos o padres en el barco la edad promedio es de 32
- para las personas que tengan un hijo o padre en el barco la edad promedio es de 24
- para las personas que tengan dos hijos o padres en el barco la edad promedio es de 17
- para las personas que tengan tres hijos o padres en el barco la edad promedio es de 33
- para las personas que tengan cuatro hijos o padres en el barco la edad promedio es de 44
- para las personas que tengan cinco hijos o padres en el barco la edad promedio es de 39
- para las personas que tengan seis hijos o padres en el barco la edad promedio es de 43

Ya una ya teniendo la medias en base las edades generamos una función que nos apoye a rellenar los datos faltantes y checamos si realmente rellenos los valores faltantes:

```python
def age_aprox(cols):
    Age = cols[0]
    Parch = cols[1]
    # Checamos si  los valores esta vacios o no existen
    if pd.isnull(Age):
        # Si es asi usamos la aproximacion de grupos de hijos o padre por
        # media de edades de los pasajeros en el barco como esta arriba
        if Parch == 0:
            return 32
        elif Parch == 1:
            return  24
        elif Parch == 2:
            return  17
        elif Parch == 3:
            return  33
        elif Parch == 4:
            return  44
        else:
            return  39 # 5 hijos o padres
    else:
        return Age

# Agregamos los datos por aproximacion
titanic_data['Age'] = titanic_data[['Age', 'Parch']].apply(age_aprox, axis = 1)
```

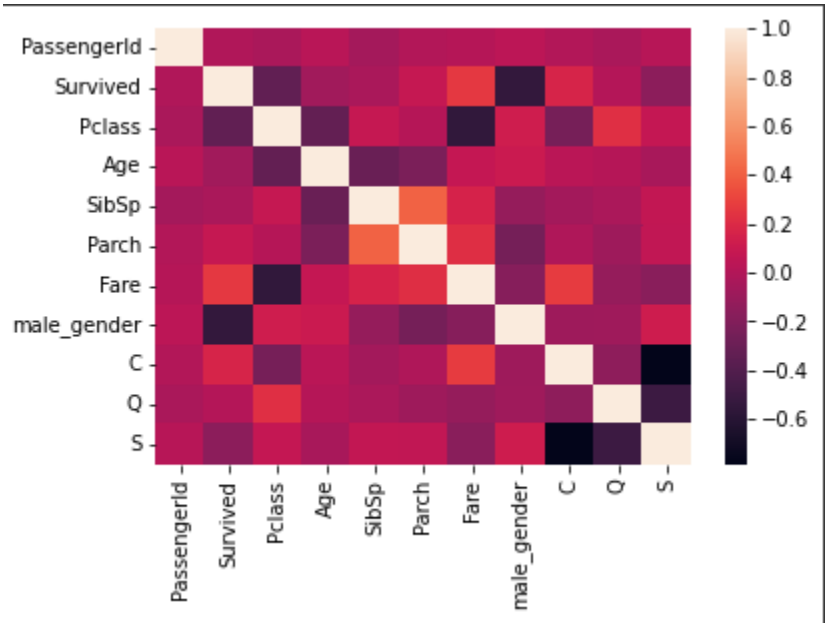
```python
# Checamos que no haya vacios (cosa ya no habra)
titanic_data.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       2
dtype: int64
```

Por otro lado generamos un Label Encoder y un One Hot Encoder para la realización de “traducir” las variables categóricas a numéricas binarias para su uso mas adelante, ejemplo, realizamos un One Hot Encoder r en la columna Embarked para la creación de diferentes columnas en base a las opciones disponibles en dicha columna y tener un valor binario por fila en caso que esta sea cierta o no de salir de cierto puerto.Finalmente la columna limpia de la siguiente manera:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	male_gender	C	Q	S
0	1.0	0.0	3.0	22.0	1.0	0.0	7.2500	1.0	0.0	0.0	1.0
1	2.0	1.0	1.0	38.0	1.0	0.0	71.2833	0.0	1.0	0.0	0.0
2	3.0	1.0	3.0	26.0	0.0	0.0	7.9250	0.0	0.0	0.0	1.0
3	4.0	1.0	1.0	35.0	1.0	0.0	53.1000	0.0	0.0	0.0	1.0
4	5.0	0.0	3.0	35.0	0.0	0.0	8.0500	1.0	0.0	0.0	1.0

Ya teniendo el dataset limpio de valores categóricos, objectos y vacíos ,checamos los valores de independencia entre los atributos:



Lo que nos dice la gráfica que es si tenemos correlación cerca a uno o uno negativo, significa que

- se obtuvo una fuerte relacion lineal entre el par de variables
- regresión logística asume que los atributos deben ser independientes del uno con el otro lo cual no podemos tener esto

- Procedemos a tumbar la columna Fare y Pclass ya que no son independientes del uno y del otro.

Lo cual nos da el resultado del siguiente dataframe:

	PassengerId	Survived	Age	SibSp	Parch	male_gender	C	Q	S
0	1.0	0.0	22.0	1.0	0.0	1.0	0.0	0.0	1.0
1	2.0	1.0	38.0	1.0	0.0	0.0	1.0	0.0	0.0
2	3.0	1.0	26.0	0.0	0.0	0.0	0.0	0.0	1.0
3	4.0	1.0	35.0	1.0	0.0	0.0	0.0	0.0	1.0
4	5.0	0.0	35.0	0.0	0.0	1.0	0.0	0.0	1.0

Ya con este arreglo en el Dataset ya podemos generar el despliegue del modelo para su división en entrenamiento, validación y prueba en base a la columna de “Survived” con la finalidad de predecir y dar a conocer que tantos pasajeros sobrevivieron en base a la misma. Esta separación se generó un set de entrenamiento de 4/5 para su aprendizaje de los datos, como tambien un set de validación de 1/5 de los datos , las cuales todos esta “aleatoriamente” acomodados con la finalidad de que encontrar una mejor predicción ,y porcentaje de precisión más adelante, como tambien evitar un overfitting con los datos y este modelo se los aprenda los pasos y datos de memoria, esto se hizo de la siguiente manera:

```
# MODEL DEPLOYMENT
# Rompemos el dataframe para  el set entrenamiento (4/5 de datos del dataframe) y  set pruebas (pass set) (1/5 de datos del dataframe)
# y quitamos la variable Survived qque es la que queremos checar
X_train, X_test, Y_train, Y_test = train_test_split(titanic_data_dmy.drop(['Survived'], axis= 1), # Valores en X
                                                    titanic_data_dmy['Survived'], test_size = 0.2, # Valores en Y
                                                    random_state= 0) # seed de random para tener los mismos resultados
```

Las cuales tiene los siguientes tamaños por set:

```
Valores de entrenamiento en X :
(711, 8)
valores en Y:
 363    0.0
255    1.0
476    0.0
271    1.0
553    1.0
      ...
835    0.0
192    1.0
629    1.0
559    0.0
684    0.0
Name: Survived, Length: 711, dtype: float64
/-----------------------------------/
```

Ya aplicando el modelo de aprendizaje máquina, la cual fue regresión logística podemos decir que:

Una vez ya realizada la primera prueba que fue:

```
# Llamando al modelo
# Model training
model = LogitRegression(learning_rate = 0.001, epochs = 100000)
model.fit(X_train, Y_train)  # tarda su rato dependiendo las epo

# Prediction on test set
Y_pred = model.predict(X_test)

print(f'Prediccion Modelo realizado de prueba 1:\n {Y_pred}')  #
print("/-----------------------------------/")

#Reporte de Clasificacion  sin cross validation del modelo gene
print(f'Reporte de clasificacion sin crossvalidation  de prueba

print("/-----------------------------------/")
```

La cual soltó estos resultados:

```
/---------------------------------/
Prediccion Modelo realizado de prueba 1:
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
/---------------------------------/
Reporte de clasificacion sin crossvalidation  de prueba 1:
              precision    recall  f1-score   support

         0.0       0.64      0.99      0.77       109
         1.0       0.88      0.10      0.18        69

    accuracy                           0.65       178
   macro avg       0.76      0.55      0.48       178
weighted avg       0.73      0.65      0.54       178
```

Ahora bien, en el modelo manual generado, podemos ver que tiene una predicción/bias del 65% la cual no está mal pero podría ser mejor, dando un sesgo “bajo”/medio al momento de obtener el valor promedio de la predicción repitiendo el proceso del modelo y llevando su proceso la cual se rige a por la cantidad de pocas , modelo, y datos que se presente dentro del mismo.

En cuanto el porcentaje del bias se debe que el modelo a los siguientes aspectos:

- El modelo tiene un margen de error de predicción grande y no se modifica con cada interacción, lo cual gran parte de los datos pasan dentro del mismo margen
- Debido a que no se tiene un margen de error “manual” y que no se modifica por cada interacción por época, esto genera que la ratio y las épocas mejoren el modelo por cada época realizada
- Las épocas y el ratio de aprendizaje se genera en base al usuario, es decir, manualmente se agregan los valores esperados para estas variables, haciendo que el modelo y la precisión resultante se vean afectadas por lo mismo.

En cuanto la varianza, en este modelo, se ajusta en base a los conjuntos de entrenamiento para asi comprender la diferencia entre las variables, sin embargo no encontramos la precisión total del mismo, pero nos encuentra las irregularidades del modelo al realizar y usar las predicciones de diferentes conjuntos de datos, ya sea con aleatorios, o no , lo cual en este caso, los datos aleatorios nos apoyan (dependiendo de la semilla que se utiliza como tambien la separación de los set de entrenamiento, de validación y el ratio de aprendizaje que se use en el modelo), la varianza ira variando en base a las pruebas realizadas, por ejemplo en la primera prueba tiene una varianza media, la cual no todos los datos están correctos pero otros si , generando un modelo con resultados correctos pero de manera inconsistente, con la tercera prueba, se genere un modelo con resultados correctos y de manera concreta y con la segunda prueba genere un modelo con resultados incorrectos y manera inconsistente.

Por estas causas, el modelo no está “efectivo” o similar que el modelo generado por la librería, ahora bien esto se puede mejorar, si tomamos en cuenta lo anterior, podemos hacer los siguiente para mejorar el modelo:

- Que el usuario únicamente tenga la opción de agregar manualmente el ratio de aprendizaje
- Tener las épocas calculadas en base la iteración de los datos con el margen de error de estos.
- Tener un margen de error mínimo, para realizar los procesos de los datos más precisos.
- Que las épocas vayan incrementadas de acuerdo con la satisfacción de la condición al de margen de error y asi obtener la mejor precisión posible de los datos.

Entre esas soluciones y otras que el momento no veo, se podría mejorar el modelo manual y aumentar su precisión y optimización, cercándolo o hacerlo similar al modelo generado por el modelo.